(2)

Autonomous Neural Network Controller for Adaptive Material Handling

ONR Contract No. N00014-91-C-0258

Quarterly Progress Report:
November 28, 1991 through February 28, 1992

Dr. James Kottas
Dr. Michael Kuperstein
Symbus Technology, Inc.
325 Harvard Street
Brookline, MA 02146

Abstract

Current methods in motor control have problems dealing effectively with highly variable dynamic inertial interactions between multijointed robots and payloads. We are developing an autonomous neural network controller that can overcome these difficulties by learning to anticipate the inertial interactions from its own experience. The neural network controller will allow robots to handle diverse payloads in uncertain environments to benefit a wide variety of material handling applications. Our target application is bin-picking, the grasping of an object from a bin containing many randomly oriented objects and placing it at a desired location.

During this quarter of the SBIR Phase II contract, we focused on the dynamic control aspect of the problem by extending our working implementation of the neural network controller from the Phase I effort. Using a commercially available scara-type robot, we demonstrated a functional prototype of the neural network controller for realizing point-to-point control. The controller design consists of dynamic position and velocity servos in parallel with an adaptive neural network controller for each joint. The controller adaptively learns to compensate for the dynamic inertial interactions with different payloads through its own experience. Using two joints of the scara robot, the controller achieved a position accuracy of 0.2% of the joint range, a timing accuracy of within 5% of the requested movement time, and an end-point stability of within 8% of the maximum planned velocity. This performance was measured on both joints after only 150 training iterations with a movement that had large dynamic coupling forces between the scara links.

1. Introduction

For typical material handling applications in automation, 80% of the cost is required for customized tooling in order to constrain both the handling process and part presentation. This tooling is necessary to maintain very low tolerances between the expected material path and the robots handling the material because current robot control technology cannot deal effectively with variations in the workplace. Controllers which allow robots to be flexible and adaptable to changes in the environment would significantly reduce the tooling cost.

For this research contract, we are focusing our development work on the problem of bin-picking. In this application, a robot must pick out a single object from a bin containing many randomly oriented objects and then place it at a desired location. For the robot to be flexible and adaptable in doing bin-picking, it must be able to pick up and move a wide variety of objects with different sizes, shapes, weights, and weight distributions. Then, the same robot could be used for most bin-picking applications without requiring any customized tooling or reprogramming.

Bin-picking can be divided into five tasks:

1. **Dynamic multijoint control.** Moving a multijoint robot arm with or without an object from one location to another introduces inertial interactions and dynamic coupling between the joints. These dynamic effects are enhanced if the weight of the object is comparable to or larger than the weight of the arm.
2. **Object singulation.** Given a bin of randomly oriented and overlapping objects, the controller must be able to single out one of the objects and determine its orientation and location in three-dimensional space.
3. **Visually-guided reaching.** Assuming the location and orientation of the object has been computed, the controller must be able to move the gripper on the arm from its current position to the object with the correct orientation in three-dimensional physical space.
4. **Sensor fusion of vision, tactile and force-feedback for gripping.** Once the gripper has been positioned very close to the object, these sensor modalities must be analyzed in conjunction with each other in order to fine-tune the process of gripping the object.
5. **Holding and releasing the object.** After the object has been gripped, the controller may need to adjust the grip in order to prevent the object either from slipping out during a fast movement or from being damaged by the gripper. In addition, the controller also must sense the actual orientation of the object in the gripper. Different orientations most likely will require different end points for the movement in order for the object to be placed at its proper location with the correct orientation.

In this organization, each task presumes the previous tasks have been completed and are available.

During this quarter, we concentrated on the first task, dynamic multijoint control. The goal of this task is to generate fast, accurate, and stable movements. However, the exact trajectory taken during the movement is not critical. Thus, we are concentrating on a point-to-point control strategy.

Current control methods in industry rely mainly on proportional-integral-differential (PID) control. Although simple, PID controllers have several shortcomings that prevent them from being useful for general material handling applications. First, they must be tuned manually for any given type of robot. This tuning results in a set of three parameters known as the PID coefficients. These coefficients are used in the PID formulation to control the robot over its entire range of movement, speed, and payload. However, the dynamic forces on the robot arm vary with position and payload. Therefore, the PID coefficients must be selected with the most common movement range and payload in mind. As a result, the performance of the robot over its full range of operation can be reduced significantly. In material handling applications involving many robots, one slow robot can create a bottleneck for the entire automation line.

In the regions of the robot range where the PID coefficients are not well tuned, the robot arm can exhibit several characteristics that further degrade the performance of the movements. With a point-to-point control strategy, the main problem in realizing fast, accurate, and stable movements is end-point oscillations. When a robot is carrying large payloads relative to its maximum payload, a fast movement implies the robotic arm will develop a relatively large amount of momentum due to the inertial qualities of the payload.[1] This momentum can cause the arm to overshoot the desired position and possibly oscillate about it. A *stable* movement is defined to be one in which the end effector of the robot does not overshoot nor oscillate about the desired final position in physical space. Similarly, an *accurate* movement is defined to be one in which the end effector stops within an acceptable tolerance of the desired final position at the right time.

The goal for this quarter was to extend our dynamic control solution for obtaining fast, accurate, and stable movements with a single joint robot to include multiple joints. The main issue to be resolved here is the dynamic coupling between the joints. For any particular joint, the dynamics of how that joint moves depends on where the other joints are and what they are doing. For N joints, this problem is N-dimensional in complexity. The solution we seek must be as simple as possible, rely on a minimum number of adaptable parameters and fixed constants, be insensitive to noise and faults, operate autonomously (hands-off), and generalize to any robot with any number of joints and degrees of freedom. Furthermore, the solution must be robust in that the adaptable parameters are guaranteed to converge to values which always satisfy the desired accuracy and stability conditions for the movement.

The robot used to develop the dynamic multijoint control formulation is a commercially available American Robot model S-1400 and is depicted in Figure 1. It is a four-axis scara-type robot with a maximum payload of 50 pounds. Only two axes were used for our initial demonstrations, the shoulder joint (J1) and the elbow joint (J2). The links associated with these joints operate in a horizontal plane. The other two joints were not powered. The shoulder joint is capable of moving at speeds of approximately 8 feet per second and has a range of 300 degrees. Furthermore, it contains a high-resolution optical encoder which provides 258,000 counts over the range for accurate position and velocity information. A 386 PC computer communicates with the robot and its drive electronics via a commercial data acquisition board (Data Translation model DT2812-A) and an encoder board (Technology 80 model 5638).

Our dynamic multijoint controller is a biologically inspired control system with adaptable parameters to govern the dynamics. The operation of the controller is feedforward at a high level and feedback at a low level. The high-level feedforward aspect is due to the fact that the governing parameters are computed *before* each movement. During a movement, these parameters are fixed and local position and velocity feedback is used to allow the movement dynamics to emerge naturally. At the end of the movement, the actual timing, accuracy, and stability are measured to form an error criterion for adapting all the governing parameters. The accuracy error is a position and time measurement and the stability error is a velocity measurement.

1. Even with no payload, a robotic arm can exhibit inertial effects when it is relatively heavy.

The dynamic multijoint controller is composed of identical local controllers for each joint that receive global position inputs from all joints. Each joint controller is responsible for moving its corresponding joint without any knowledge of the control signals being sent to the other joints. The inputs to each joint controller are the desired movement time, the initial and desired final positions for all joints, and a subjective measure of the current payload. The neural networks used in each joint controller to store the adaptable parameters are structured to reduce considerably complexity of the dynamic coupling problem without any loss of generality. The dynamic multijoint controller is described in more detail in Section 2.

The approach of measuring the error information once for each movement is in contrast to current adaptive methods for controlling robots. These methods are much more complicated than the prevalent PID controllers. They involve one or more optimal criteria and need to update the governing parameters *during* the movement to achieve the criteria. As a result, these methods require much more computation power in order to sample the state of the robot's joints quickly. In contrast, our approach is significantly less demanding computationally during the movement. Instead, our high-level feedforward solution performs most of the computation before the movement is started. The low-level feedback components (the local joint controllers) involve only simple feedback and calculations at the joint.

In addition, the optimal adaptive control formulations are global methods in that the next motor signal a joint receives during a movement depends upon the current states of *all* joints. Our joint controllers are local and do not depend upon the other joints during a movement. A joint controller only uses the initial and final positions of all joints before a movement to calculate the current set of governing parameters.

Our experimental results with the dynamic multijoint controller are discussed in Section 3. We implemented a simplified version of the controller which did not take into account variable payloads at this time because we did not have any way to subjectively measure them during this quarter. The results show that the controller learns to make accurate and stable movements very quickly. The adaptable governing parameters converge onto their optimal values, albeit more

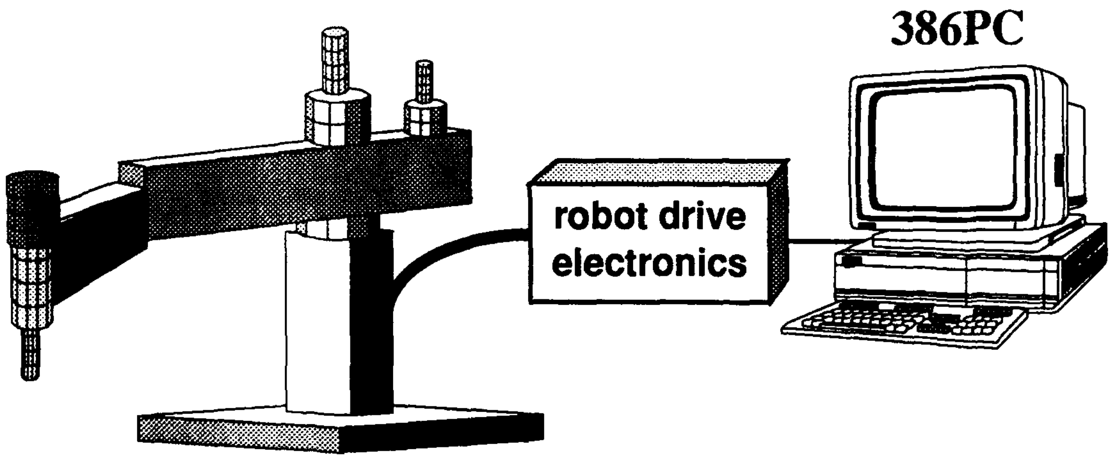

Figure 1: Schematic of the scara robot and its associated control hardware used to develop the dynamic multijoint control formulation of our neural network controller.

slowly than the movement performance would suggest. This characteristic indicates the robustness of the controller design.

2. Dynamic Multijoint Control Module

For a robot with N joints, the dynamic multijoint control module in the neural network controller is composed of N local joint controllers, one of which is shown in Figure 2. Each local joint controller operates in two modes, posture and movement. The role of the posture mode is to keep the arm at its desired position, irrespective of gravity and payload. If a new desired position is set, the posture mode will move the arm there with accuracy but not on time or with stability. Movement mode is responsible for establishing the on-time arrival and the end-point stability. These modes operate in parallel (additively) with the movement mode being activated when the desired position changes.

The local joint controller has four inputs:

1. The initial position of each joint in the arm in joint space (as opposed to physical space). These angles will be denoted by x_{i0}.
2. The desired position for each joint in joint space, denoted by x_{id}.
3. The desired movement time, T_d.
4. A measure of the current payload, P_L. This measure can be subjective and only needs to be repeatable and monotonic with the payload's actual weight.

The first two inputs are used directly by the local joint controller and the second two inputs are used

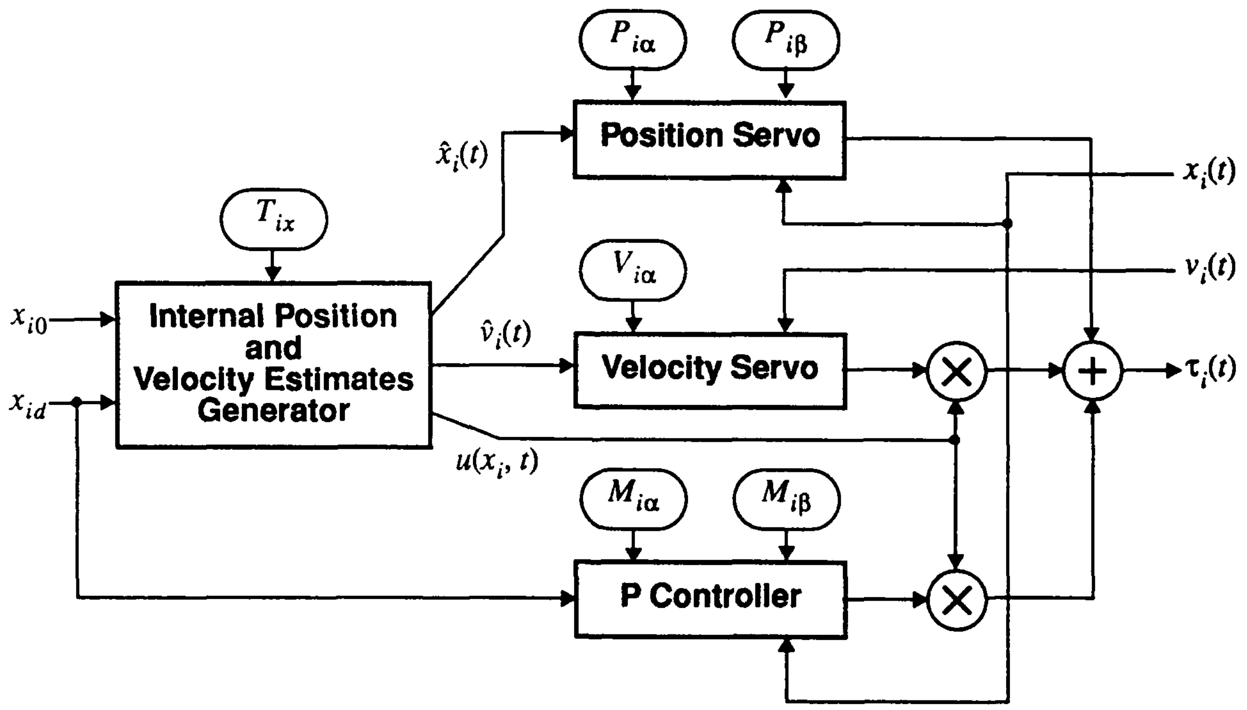

Figure 2: Block diagram of a local joint controller. For a robot with N joints, the complete dynamic multijoint control module of the neural network controller consists of N distinct local joint controllers. The adaptable parameters are enclosed by horizontal ovals.

only to compute parameter values.

To perform a movement, an internal estimate of the position and velocity is generated for the i^{th} joint according to

$$\frac{d}{dt}\hat{x}_i(t) = T_{ix}\operatorname{sgn}(x_{id} - x_{i0})\, u(x_i, t), \tag{1}$$

$$\hat{v}_i(t) = \left(\frac{[\hat{x}_i(t) - x_{i0}]\,[x_{id} - \hat{x}_i(t)]}{6\,(x_{id} - x_{i0})^2}\right) u(x_i, t), \tag{2}$$

where $\hat{x}_i(t)$ is the position estimate, $\hat{v}_i(t)$ is the velocity estimate, and T_{ix} is the position integration rate. The timing of the movement is governed by T_{ix}, which in turn depends on the desired movement time. The function $u(x_i, t)$ is the local movement signal defined by

$$u(x_i, t) = \begin{cases} 1 & \text{for } x_{i0} \le \hat{x}_i(t) < x_{id} \text{ if } (x_{id} \ge x_{i0}), \\ 1 & \text{for } x_{id} \le \hat{x}_i(t) < x_{i0} \text{ if } (x_{id} < x_{i0}), \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

This function is 1 when the movement mode parameters are to be active and 0 when only the posture parameters are in control.

The true velocity estimate $\frac{d}{dt}\hat{x}_i(t)$ is simply a constant value over some period of time. However, this form does not acknowledge the inertial properties of the arm because it presumes a step change in velocity is possible. The form for $\hat{v}_i(t)$ given above is parabolic and has a symmetrical bell-shaped profile. This form is more realistic because it only presumes a step change in acceleration is possible. The scale factor is needed in the parabolic $\hat{v}_i(t)$ so that the velocity estimates integrates over time to be $x_{id} - x_{i0}$, the net change in position.

The position estimate is then used as the reference for a position servo, described by

$$\tau_{i,\,\mathrm{Pos}}(t) = P_{i\alpha}\,[\hat{x}_i(t) - x_i(t)] + P_{i\beta} \tag{4}$$

where $x_i(t)$ is the current position of the joint, $P_{i\alpha}$ is the position servo gain, and $P_{i\beta}$ is a constant bias to compensate for external forces such as gravity. This torque term constitutes the posture mode.

The movement mode is composed of a velocity servo and an open-loop P controller. The velocity servo is driven by the velocity estimate according to

$$\tau_{i,\,\mathrm{Vel}}(t) = V_{i\alpha}\,[\hat{v}_i(t) - v_i(t)], \tag{5}$$

where $v_i(t)$ is the current velocity of the joint and $V_{i\alpha}$ is the velocity servo gain. While the position and velocity servos operate relative to the estimated position and velocity, respectively, the open-loop P controller is a simple spring-like restoring force that is relative to an absolute reference, the desired final position x_{id}:

$$\tau_{i,\,\mathrm{Mov}}(t) = M_{i\alpha}\,[x_{id} - x_i(t)] + M_{i\beta}, \tag{6}$$

where $M_{i\alpha}$ is the gain (analogous to the stiffness of the spring) and $M_{i\beta}$ is a bias offset (the rest position of the spring relative to the scaled desired final position). This control formulation is called "open-loop" because the reference position is fixed during the movement.

The total torque signal that is sent to the motor amplifier, $\tau_i(t)$, is the sum of all the individual torques with the movement mode terms being gated by the movement signal function:

$$\tau_i(t) = \tau_{i,\,\mathrm{Pos}}(t) + [\tau_{i,\,\mathrm{Vel}}(t) + \tau_{i,\,\mathrm{Mov}}(t)]\,u(x_i, t). \tag{7}$$

The role of each term has physical significance. The position servo maintains the final position accuracy but not the timing nor the stability. When the movement mode terms are active, the open-loop P controller provides on-time arrival and stability. The velocity servo provides coordination between the joints to compensate for the dynamic coupling forces to increase the stability of the coupled joints. It is one way to deal with these forces, but not the only method.

At the end of the movement phase (signaled by $u(x_i, t)$ going from 1 to 0), the performance of the movement is observed by each local joint controller. Three measurements currently are available:

1. The time when the movement phase ended, denoted by T_i.
2. The position of the joint, x_{im}.
3. The velocity of the joint, v_{im}.

If the movement had some instabilities, the arm may still be in motion for some small amount of time after the movement phase ended. When the arm finally stops moving (under the sole influence of the posture mode position servo), a fourth measurement is available: the final position of the joint, x_{if}. These measurements translate into the following error quantities for each local joint controller:

1. The arrival time error, $\Delta T_i = T_d - T_i$.
2. The movement position error, $\Delta x_{im} = x_{id} - x_{im}$.
3. The movement velocity error, $\Delta v_{im} = v_{id} - v_{im}$ which equals $-v_{im}$ since $v_{id} = 0$.
4. The posture position error, $\Delta x_{if} = x_{id} - x_{if}$.

These errors can be used to adapt the parameters of the local joint controller.

In the most general form, the adaptable parameters are T_{ix}, $P_{i\alpha}$, $P_{i\beta}$, $V_{i\alpha}$, $M_{i\alpha}$, and $M_{i\beta}$.[2] Although several error functions are possible, the simplest ones that provided the fastest convergence are:

$$\delta_{T_{ix}} = \Delta T_i, \tag{8}$$

$$\delta_{P_{i\alpha}} = \begin{cases} \Delta x_{if} & |\Delta x_{if}| > \varepsilon_f \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

2. If desired, the scaling factor for the velocity estimate $\hat{v}_i(t)$ could be made adaptable in the manner of T_{ix}.

$$\delta_{P_{i\beta}} = \Delta x_{if}, \tag{10}$$

$$M_{i\alpha} = -\Delta v_{im}, \tag{11}$$

$$M_{i\beta} = \Delta x_{im}, \tag{12}$$

where ε_f is the error tolerance on the final position accuracy. During our experiments, $V_{i\alpha}$ was held constant for simplicity so no error functions were explored yet.

Each adaptable parameter is encoded by a neural network. These error values are used to update the weights in the respective networks. The basic network model used by all parameters is a topographical map that is excited by a fixed bell-shaped activation function centered at the inputs to the map. The output of the map is the sum of the weighted outputs of the map. This structure is best illustrated using an example.

The position estimate integration rate (T_{ix}) depends only on the desired movement time (T_d) and the initial (x_{i0}) and final (x_{id}) positions of the movement. Since these three inputs are independent, the corresponding map for T_{ix} needs to have three dimensions. Let W represent the map so W_{jkl} denotes the neural weight at $x_{i0} = j$, $x_{id} = k$, and $T_d = l$. Furthermore, let the bell-shaped activation function be the three-dimensional Gaussian distribution,

$$g(j, k, l) = C \exp\left[-\frac{(j - x_{i0})^2 + (k - x_{id})^2 + (l - T_d)^2}{2\sigma^2}\right] \tag{13}$$

where C is a normalization constant and σ is the half-width. The map output is computed using

$$T_{ix} = \sum_j \sum_k \sum_l W_{jkl}\, g(j, k, l). \tag{14}$$

Given the error signal $\delta_{T_{ix}}$, the map is adapted using

$$\Delta W_{jkl} = \eta \delta T_{ix} g(j, k, l) \tag{15}$$

where ΔW_{jkl} is the change in the weight at index position (j, k, l) and η is the learning rate.

The maps for the other parameters are slightly more complex in that there are more inputs. For example, the posture parameters $P_{i\alpha}$ and $P_{i\beta}$ depend upon the current payload P_L and the final positions of *all* joints. The movement parameters $M_{i\alpha}$ and $M_{i\beta}$ depend upon the movement time, the current payload, and the initial and final positions of all joints. In general, the maps for parameters like $M_{i\alpha}$ and $M_{i\beta}$ could $(N + 2)$-dimensional. However, it is not necessary to relate the initial position of joint i (x_{i0}) with the final position of joint n (x_{nd}). It is important, though, to relate the movement of joint i (x_{i0}, x_{id}) with the movement of joint n (x_{n0}, x_{nd}). Thus, the $(N + 2)$-dimensional maps can be reduced to a set of N two-dimensional maps that relate x_{i0} and x_{id} for each joint and then have a separate two-dimensional map for the time and payload inputs. The output of all these maps can be summed together to form the desired parameter such as $M_{i\alpha}$. The error value $\delta_{M_{i\alpha}}$ is then applied to all the component maps in the normal way.

3. Performance Results and Discussion

Since the shoulder and elbow joints of our scara robot operate in a plane, there are no external forces such as gravity acting on the arm. Therefore, $P_{i\beta}$ is fixed at zero for simplicity. Furthermore, the position and velocity servo gains can be kept constant. However, because both joints have significant friction and drag, the position servo gain $P_{i\alpha}$ was changed from a constant to the deterministic function $P_{i\alpha}(\hat{x}_i(t) - x_i(t))$, where

$$P_{i\alpha}(x) = \begin{cases} 10 - 160|x| & \text{for } 0 \le |x| < 0.05, \\ 2 & \text{for } |x| > 0.05. \end{cases} \tag{16}$$

The lower gain away from the desired position prevented oscillatory instabilities from appearing during the movement (which also survive at the end of the movement). Conversely, the higher gain near the desired position produced good position accuracy in the presence of the friction and drag. If $P_{i\alpha}(x)$ was to become adaptable, only the peak value at $P_{i\alpha}(0)$ needs to be adapted.

The velocity servo gain was fixed at $V_{i\alpha} = 0.5$. With $P_{i\alpha}$, $P_{i\beta}$, and $V_{i\alpha}$ either fixed or functional, the adaptable parameters were T_{ix}, $M_{i\alpha}$, and $M_{i\beta}$. These parameters are the key ones for governing the timing and the stability of the movement. The learning rates for all their error values were set to 0.5.

The arm was trained to going between two locations for 300 iterations (150 iterations per movement). The results from one of these movements are shown in Figures 3-5. The results from the other movements are similar. These two positions were chosen because they are representative of the cases when the dynamic coupling forces between the links of the arm are the strongest. The robot was trained successfully to move between multiple points to demonstrate the matrix idea of maps for the movement parameters. For these movements, the desired movement time was set to $T_d = 1$ second and the payload was fixed at $P_L = 15$ pounds.

The actual movement in physical space is illustrated in Figure 3. The path shown here is the one taken by the arm after training was stopped. Note that the path is smooth but not linear in physical space (and neither is it linear in joint space). This is quite satisfactory for the point-to-point control strategy. The dynamic coupling forces are large in this case because the shoulder joint actually helps to over-accelerate the elbow joint at the beginning of the movement. When the shoulder starts to decelerate, it over-decelerates the elbow joint. For a smooth movement, the elbow joint must do some braking at the beginning of the movement and some accelerating towards the end. If not, the elbow joint will burst too greatly which induces oscillations in the elbow's trajectory. While the path is not of concern, these oscillations cannot be compensated for simply by the open-loop P controller. The velocity servo loop provides the necessary compensation.

The temporal evolutions of the trajectory for both joints is shown in Figure 4(a). All joint positions, velocities, and torques have been normalized to their respective maximum ranges. The torque profile for the shoulder joint reveals the intuitive situation whereby there is an initial burst of

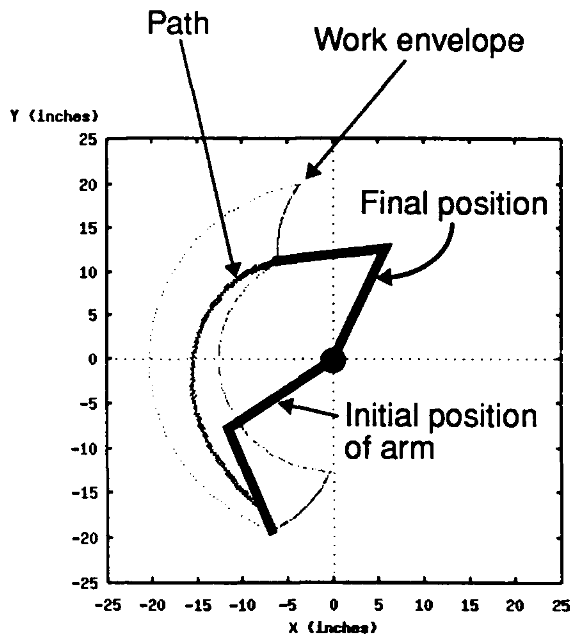

Figure 3: The trajectory of the sample arm movement in physical space after training. The center of the robot is at (0,0).

positive torque to get the shoulder moving and then a gradual decrease to some negative torque to provide the necessary braking. The effect of the open-loop P controller is to skew the velocity profile slightly from its symmetrical velocity estimate, producing a shorter burst and a longer brake. On the elbow joint, the effect of the coupling can be seen by the significant difference between the estimated and actual velocity curves. The initial velocity burst is due to the coupled acceleration of the shoulder joint. Similarly, the accelerating torque towards the end of the movement is compensating for the coupled braking force from the shoulder.

The end points at $t = 1$ second show no noticeable position error but do have some finite velocity error. The training evolution of these errors for both joints is shown in Figure 4(b). The position error quickly goes to zero and averages within 0.2% of the total movement distance for both joints. The velocity error, on the other hand, converges more slowly. It comes within 4% of the maximum estimated velocity for the shoulder joint and within 8% for the elbow joint. In order to increase the convergence speed, a larger learning rate could be used with the velocity error. In addition, a velocity error threshold could be introduced that sets the maximum tolerable velocity error, below which a zero error value would be generated.

The training evolution of the adaptable parameters for both joints is shown in Figure 4(c). Because the velocity error had not converged to zero yet, the parameters have not converged completely, but they are visually approaching asymptotes.

The evolution of the total training error is shown in Figure 5. This quantity, denoted by E_t, is defined to be the sum of all the error values over all adaptable parameters and all joints. It is a measure of how quickly a particular movement has been learned from an external performance viewpoint (rather than an internal parameter viewpoint). For the sample movement,

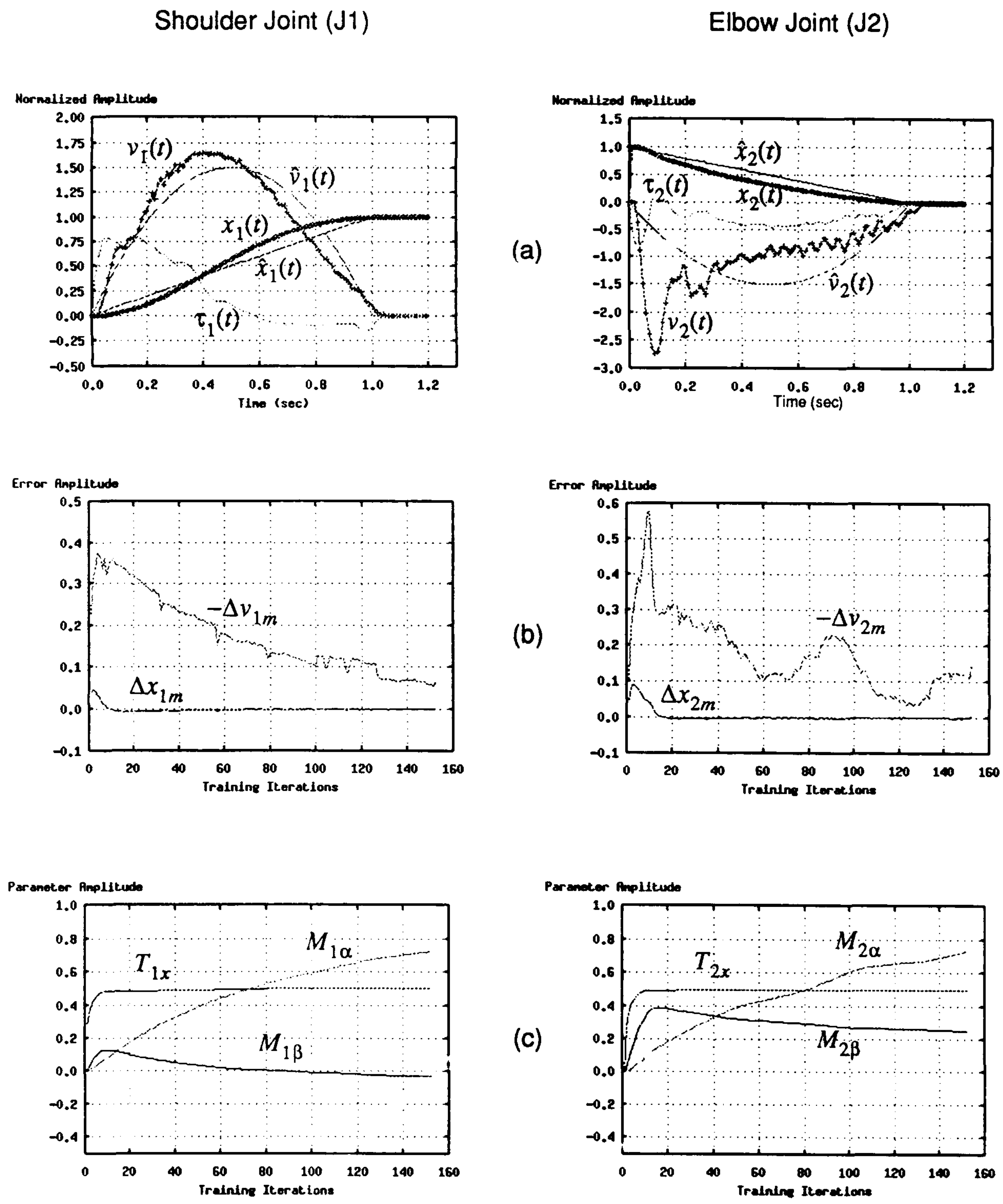

Figure 4: (a) The time evolution of the sample movement trajectory for both joints. (b) The training evolution of the end-of-movement position and velocity errors for both joints. (c) The training evolution of the adaptable parameters.

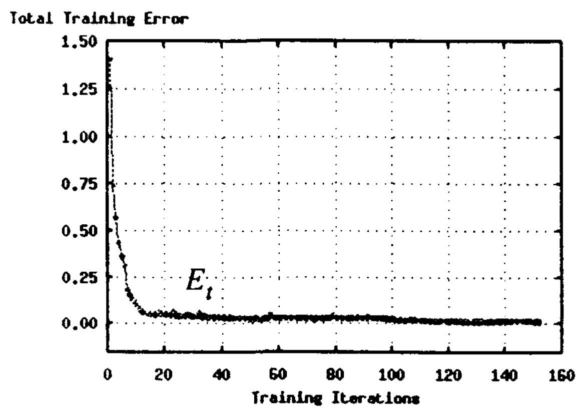

Figure 5: The evolution of the total training error.

$$E_t = \sum_{i=1}^{2} \left[\left|\delta_{T_{ix}}\right| + \left|\delta_{M_{i\alpha}}\right| + \left|\delta_{M_{i\beta}}\right| \right] . \tag{17}$$

Note that even though the parameters have not converged fully yet, the total training error decreases rapidly initially and then gradually thereafter. After the initial decrease, the movement visually appears to be quite acceptable at the end point. During the remaining gradual error decrease, the parameters are being adjusted to their final values by fine-tuning the movement's performance. In practical terms, the final velocity goes to zero quickly after 1 second. The actual movement takes about 1.05 seconds, resulting in an effective timing error of about 5%.

These plots demonstrate our initial success with the dynamic multijoint control module. However, several compromises were made to simplify the model. First, the velocity servo gain was a constant. Ideally, it should be adaptable but no suitable error function in terms of the end-point error measurements could be found. Since the velocity servo compensates for extreme velocities during the movement, it effectively is modifying the path. Since we have no direct path error information, the error function could be some integral measure of a path quantity that is observed only at the end point.

Another compromise was that a fixed payload was used. Since no force sensors were available on the robot at the end effector, no measurement of the payload was possible. Similarly, a single movement time was considered for each movement (although it could vary between different movements). However, our goal is to handle any payload and any physically realizable movement time.

To realize these capabilities with the dynamic multijoint control formulation, a two-pass training approach can be used and the adaptable parameters can have additional maps with P_L and T_d as their inputs. The output of these maps can be either additive or multiplicative with the total output from the spatially-indexed maps (those whose inputs are the initial and final joint positions). During the first pass, both the payload and the movement time are fixed. The robot is trained to move between any number of locations and the spatially-indexed maps are updated with the appropriate error values. Then, a different payload and/or movement time is chosen and the training is repeated between the same positions. However, the spatially-indexed maps are held constant this time and only the maps indexed by payload and movement time are updated with the corresponding error values.

4. Future Work

Given our initial success with the dynamic multijoint control module of the autonomous neural network controller, we plan to begin the second phase of the bin-picking problem (visually-guided reaching). This process involves the following steps for the next quarter:

1. Extending the dynamic multijoint control module to handle six degrees of freedom and demonstrating dynamic control again. This step provides continuity between robots and allows us to investigate error functions for making the velocity servo gain adaptable. In addition, the revolute arm will be affected by gravity, requiring the posture bias ($P_{i\beta}$) to be adapted. It also allows us to add variable payloads and timings to the dynamic control module.
2. Integrating the concepts associated with reaching in three dimensions as learned by the INFANT model [Kuperstein 1991].

References

Kuperstein, M. (1991) INFANT Neural Controller for Adaptive Sensory-Motor Coordination, *Neural Networks*, V4 pp. 131-145.